REMARKS

Claims 1-18 are pending in the current application. Applicants have amended claims 1, 4, 5, 8, and 10-15 to correct minor informal typographical errors and claim numbering issues, and not for reasons related to patentability. Applicants appreciate the identification of these clerical inaccuracies so they could be addressed. Reexamination and reconsideration of all claims are respectfully requested.

Applicants acknowledges and appreciates the indication of allowable subject matter for claims 5, 7, 12, 13, and 16-18.

SPECIFICATION

The Office Action objected to the disclosure because of an informality at line 15 of page 15 of the Specification. Applicants appreciate this typographical issue identification and have corrected the cited passage in the Specification.

CLAIM OBJECTIONS

The Office Action objected claims 1-8, 10-14, and 16 because of various informalities. Applicants appreciate these being brought to Applicants' attention and have modified the claims as follows:

a. With respect to claim 1, Applicants have amended the claim to remove the letter/word "a."

b. With respect to claims 5-8, Applicants have amended the claim numbering.

c. With respect to claim 10, Applicants have amended the claim to fix a typographical error and indicate dependency from claim 9.

d. With respect to claim 11, Applicants have amended the claim to fix a typographical error and indicate dependency from the proper claim.

e. With respect to claims 12 and 13, Applicants have amended the claims to fix a typographical error and indicate dependency from the proper claim.

f. With respect to claim 14, Applicants have amended the claim to fix a typographical error to include the word "by."

g. With respect to claims 12 and 16, Applicants has amended the claims to fix a typographical error and correct the status bit wording.

h. With respect to claims 2-4, Applicants have reordered claims 4 and 5.

35 U.S.C. § 112

The Office Action rejected claims 8, 13, 15-18 based on being indefinite. The Office Action at page 4 objects to claim 8 as having insufficient antecedent basis. Applicant has changed claim 8 to provide antecedent basis. The Office Action also objected to claim 13 as having insufficient antecedent basis. Applicant has changed the dependency of claim 13. Applicant has reworded claim 15 for clarity.

Claims 16-18 are rejected because of their dependencies. Applicants contend that the dependencies of claims 16-18 are correct. Should the Examiner maintain this rejection, Applicants request further information about the alleged improper dependencies to fully respond to the rejection.

Applicants therefore submit that claims 8, 13, and 15- 18 are fully supported and offer proper dependencies and that and all claims conform to 35 U.S.C. § 112.

35 U.S.C. § 102

The Office Action rejected claims 1-4 and 9-11 under 35 U.S.C. § 102(e) as being anticipated by Kulkarni et al., U.S. Patent 6,775,423 ("Kulkarni"). Applicants respectfully traverse the rejections of these claims.

Kulkarni discloses a method and system for incrementally updating a first image in flash memory of a device by downloading a differences file that identifies differences between the first image and a second image and applying the differences file to the first image to create the second image in the flash memory. (Abstract) Kulkarni goes on to further describe the download a modified differences file using an incremental dial-up bootloader. Kulkarni spells out in the passage at Colum 9 lines 59-65:

> FIG. 6 illustrates the process performed on the PDA 104 to incrementally update the image 139 in flash memory 132. At step 600, *the PDA 104 boots into the incremental dial-up bootloader 134 and dials into the server 102 to download the modified differences file 124.* The modified differences file 124' is downloaded to the PDA 104 and stored in the RAM 130 at step 402. (Emphasis added)

Thus, Kulkarni design requires the device requiring an update to be connected to an update server via a dial-up modem attached to a physical network in order to obtain the modified differences file.

The present design provides a system and method for adding new functionality or resolve problems found after deployment of a mobile wireless device, such as a mobile phone, without being recalled by a manufacturer for modification at a service centre. The present design provides an arrangement that allows the update package to be provided via over-the-air (OTA) delivery and for the device to reliably apply the update itself even when the wireless connection or power is lost.

The present design is more involved and fundamentally different from the Kulkarni design. Applicants' device and claims operate in a fundamentally different manner from the design description provided in Kulkarni.

Claim 1

Claim 1 of the present application includes two limitations: an update generator that produces an update package resulting from a comparison between the first data image and the second data image whereby said comparison selects and encodes an instruction set comprising a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks; and an update decoder resident on the client device, whereby said update decoder interprets the instruction set of the update package and applies the update a package to update the k memory blocks.

The present Office Action relies on column 4, lines 11-50 of Kulkarni in rejecting the "An update generator that produces an update package resulting from a comparison between the first data image and the second data image" limitation of claim 1. (Office Action, p. 5) The entire limitation is "a comparison selects and encodes an instruction set comprising a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks." The cited column 4 passage of Kulkarni states:

> The server 102 includes a processor 106, memory 108 and a network interface 110 for communicating with remote computing devices over a network, such as the Internet. The memory 108 includes an operating system 112, an old image 114 and a new image 116. The old image 114 is an execute-in-place (XIP) image to be stored in and run from flash memory. The old image 114 may be any image that can be stored in flash memory, but for purposes of the present discussion, the old image 114 is an operating system for the PDA 104.
>
> The new image 116 is an updated version of the old image 114. In the present example, the new image 116 is a later version of the operating system represented by the old image 114. The new image 116 contains sections of data that are identical to sections of data in the old image 114. The new image 116 also contains sections of data that are not included in the old image 114.

> The memory 108 also includes a compression module 118 configured to create a differences file 120 from the old image 114 and the new image 116. The differences file 120 is transmitted by the server 102 to the PDA 104. Therefore, compressing the differences file 120 prior to transmitting it to the PDA 104 significantly reduces the time required to transmit the differences file 120 to the PDA 104. *The compression module 118 constructs the differences file 120 by including the new data sections contained in the new image 116. For the data sections in the new image 116 that are also contained in the old image 114, the compression module 118 includes references in the differences file 120 to the data sections in the old image 114.* Since a reference is typically much smaller than a data section it references, the differences file 120 is significantly smaller than the new image 116.
>
> A reordering module 122 is also included in the memory 108. The reordering module 122 is configured to identify data sections in the differences file 120 that require data from data sections in the old image 114 to be created. Such data sections are called dependent data sections because they depend on other sections--particularly, referenced data sections--for constructions.

(Emphasis Added)

This passage simply states that the Kulkarni design uses a compression module to construct a 'difference file' by including the new data sections contained in the new image and including references to retained data sections in the old image. No selecting and encoding an instruction set comprising a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks is described by Kulkarni, as required by the present claim 1. The present design instruction set comprises a specific set of operations, namely SETBLOCK operations that direct updating of the memory blocks in an order that specifically employs the COPY and ADD operations required and resulting update package size. As noted in the

Specification and as the SETBLOCK operation is understood, "[t]he SETBLOCK operation identifies the next block to be updated to the decoder; although not technically required if the direction of processing is known and all blocks are updated, it allows for further optimizations as noted later, for example the ability to remove unchanged blocks from the update package entirely, and also remove trailing COPY operations from a block..." Specification, p. 29, ll. 3-9. SETBLOCK operation and SETBLOCK functions are described and employed throughout the present Specification. Kulkarni does not use such a SETBLOCK operation to direct the COPY and ADD operations required in order to reduce the size of the update package. No similar operation is disclosed. For this reason alone, namely absence of SETBLOCK, COPY, and ADD operations, all independent claims are allowable over the Kulkarni reference.

The present Office Action relies on column 5, lines 51-59 of Kulkarni in rejecting the "whereby said comparison selects and encodes an instruction set comprising a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks" limitation of claim 1. (Office Action, p. 5) The cited column 5 passage of Kulkarni states:

> The differences file 120 and the modified difference file 124 are composed of two alternating types of sections: data sections contain data from the new image 116 that is not found in the old image 114; copy sections contain data that is found in both the new image 116 and the old image 114. To distinguish between the two types of sections, a header is placed at the beginning of each section, the header giving specific information about the section that follows the header.

(Emphasis Added)

This passage merely states that the Kulkarni design 'difference file' contains two sections or types of information, one section type being new image data to be included and the other section type being old image data to be copied in order to construct the new image. Kulkarni also describes using a header to provide specific

information about the section that follows the header. Again, no selecting and encoding an instruction set comprising a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks is described by Kulkarni, as required by the present claim 1. The present design instruction set comprises SETBLOCK operations that direct updating of the memory blocks in an order that optimizes the COPY and ADD operations required and resulting update package size. Kulkarni does not use a SETBLOCK operation to optimize the COPY and ADD operations required in order to reduce the size of the update package.

The present design SETBLOCK operation identifies the next block to be updated to the decoder and allows further optimizations for example the ability to remove unchanged blocks from the update package entirely, and also remove trailing COPY operations from a block.

In essence, Kulkarni describes a method for using a header to distinguish between "alternating types of section". Kulkarni constructs the "difference file" and then uses a "reordering module" as described in the passage at Col. 4 lines 44 – 50:

> A reordering module 122 is also included in the memory 108. *The reordering module 122 is configured to identify data sections in the differences file 120 that require data from data sections in the old image 114 to be created*. Such data sections are called dependent data sections because they depend on other sections--particularly, referenced data sections--for constructions.

(Emphasis added)

This passage states that the Kulkarni design reorders all 'difference file' sections as needed prior to transmitting to the device requiring update since Kulkarni's arrangement only supports one direction of update processing. No SETBLOCK operation is described by Kulkarni, as required by the present claim 1.

The present Office Action relies on column 5, lines 20-25 of Kulkarni in rejecting the "update decoder interprets the instruction set of the update package and

applies the update package to update the k memory blocks" limitation of claim 1. (Office Action, p. 5) The cited column 5 passage of Kulkarni states:

> The flash manager 136 is configured to collect data sections for the image 139 in the new memory block 138 in RAM 130. When the new memory block size reaches one-half the minimum block size, the data sections comprising the new memory block are written to the flash memory 132 to update the operating system 139.

This says that the Kulkarni flash manager collects data sections for the image in the new memory block. Again, since Kulkarni does not use SETBLOCK, COPY and ADD instructions in creation of their 'difference file', Kulkarni design cannot interpret the instruction set of the update package as required by the present claim 1.

In summary, none of these Claim 1 limitations is present in Kulkarni. Kulkarni provides a system and method for a compression program responsible for generating a difference file and a second program for reordering the sections of the generated difference file to allow flash memory to be reprogrammed without first having to reconstruct the complete new image in RAM before programming it to flash memory. No interpretation of Kulkarni conforms to the express wording of claim 1. Kulkarni does not select and encode an instruction set comprised of a plurality of SETBLOCK, COPY and ADD operations for each k memory blocks as required by the express language of claim 1, nor a decoder for interpreting the instruction set of the update package in an order specified by the SETBLOCK operations according to the limitations provided in claim 1.

Kulkarni therefore does not anticipate claim 1. Further, all claims depending from claim 1 are allowable as they include limitations not shown in the cited references and as they depend from an allowable base claim.

Claim 9

Claim 9 includes similar "select and encode instruction set," "SETBLOCK, COPY, and ADD" operations, and "interprets the instruction set of the update package" limitations as discussed above with respect to claim 1. As with claim 1,

none of these limitations are present in Kulkarni. As noted above, the Kulkarni reference does not encode an instruction set comprised of a plurality of SETBLOCK, COPY and ADD operations for each of the k memory blocks, a clear missing element that in and of itself renders the claims novel over Kulkarni.

The present Office Action relies on column 9, lines 42-58 and column 9, line 66 through col. 10 line 1 of Kulkarni in rejecting the "applying the instruction set by interpreting the instruction set to direct the updating of the memory blocks in an order specified by the SETBLOCK operations" limitation of claim 9. (Office Action, p. 8). The cited column 9 passage of Kulkarni states:

> Referring back to FIG. 2d, the modified differences file 124 that results from reordering the differences file 120 shown in FIG. 2c is illustrated. For this example, it is assumed that section 120d (`xyz`) of the differences file 120 requires data from section 120b (`abc`) of the differences file 120 to be constructed.
>
> The modified differences file 124 includes a header 124a and its corresponding section 124b (`xyz`); header 124c and its corresponding section 124d (reference to `abc`); header 124e and its corresponding section 124f (reference to `def`); header 124g and its corresponding section 124h (`pqr`); and header 124i and its corresponding section 124j (reference to `jkl`). Since section 124b requires data referenced by section 124d, section 124b appears before 124d in the modified differences file 124. This ensures that the data section referenced by section 124d will be available when section 124b (`xyz`) is constructed.

The cited column 9-10 passage of Kulkarni states:

> The flash manager 136 uses the modified differences file 124' and the image 139 in flash memory 132 (the old image) to start building the new image.

The Kulkarni design employs a reordering module within the server to convert and order the data sections and references so that any data dependent data section proceeds, in the order, any referenced data section from which it requires data. In order to distinguish between two types of sections (sections containing data from the new image not found in the old image, and data that is common to both new and old images), a header is placed at the beginning of each section, the header giving specific information about the section that follows the header. Thus, as best that can be understood from the Kulkarni disclosure is that the sections are processed and reordered on the server side and during this processing and reordering header information is inserted as needed to provide specific information regarding the sections that follow. No SETBLOCK operation is described by Kulkarni, as required by the present claim 9. Again, the present design uses a SETBLOCK operation arrangement to encode information into the update package relating the order the package blocks should be decoded. When the decoder encounters a SETBLOCK operation, which identifies the next block to be updated to the decoder and allows further optimizations for example the ability to remove unchanged blocks from the update package entirely, and also remove trailing COPY operations from a block.

In summary, none of these Claim 9 limitations is present in Kulkarni. Kulkarni provides a system and method for a compression program responsible for generating a difference file and a second program for reordering the sections of the generated difference file to allow flash memory to be reprogrammed without first having to reconstruct the complete new image in RAM before programming it to flash memory. No interpretation of Kulkarni conforms to the express wording of claim 9. Kulkarni does encode an instruction set comprised of a plurality of SETBLOCK, COPY and ADD operations for each k memory blocks as required by the express language of claim 9, nor updating the memory blocks in an order specified by the SETBLOCK operations according to the limitations provided in claim 9. Kulkarni therefore does not anticipate claim 9. All claims depending from claim 9 are allowable as they include limitations not shown in the cited references and depend from an allowable base claim.

Accordingly, it is respectfully submitted that all pending claims fully comply with 35 U.S.C. § 102.

35 U.S.C. § 103

The Office Action rejected independent claim 15 under 35 U.S.C. §103(a) as being obvious based on Kulkarni in view of Estakhri et al., U.S. Patent 5,485,595 ("Estakhri"). The Office Action further rejected claims 6, 8, and 14 based on Kulkami in view of Miller '520. Applicants respectfully traverse the rejections of these claims.

Claim 15 of the present application includes three limitations: an update package including an instruction set, which instruction set comprises a plurality of ADD and COPY operations associated with each of the plurality of memory blocks to be updated; a status array comprised of a least two switchable status identifiers associated with each of the plurality of memory blocks, wherein one X of k to updated as instructed by the instruction set contained in the update package; and an update decoder resident on said client device that interprets the update package and applies the instruction set to update the plurality of blocks on a block-by-block basis, and which update decoder accesses and manipulates the status identifiers when applying said instruction set.

The present Office Action again relies on column 5, lines 51-59 of Kulkarni in rejecting the "wherein one X block of k blocks to be updated as instructed by he instruction set contained in the update package" limitation of claim 15. (Office Action, p. 10) The entire limitation is "a status array comprised of a least two switchable status identifiers associated with each of the plurality of memory blocks, wherein one X of k to updated as instructed by the instruction set contained in the update package". As noted above, the cited column 5 passage simply states that the Kulkarni design 'difference file' contains two sections or types of information, one section type being new image data to be included and the other section type being old image data to be copied in order to construct the new image. Kulkarni also describes

using a header to provide specific information about the section that follows the header.

No status array comprised of at least two switchable status identifiers associated with each of the plurality of memory blocks is described by Kulkarni, as required by the present claim 15. Moreover, the office action admits, see page 11, that Kulkarni fails to explicitly teach (1) a status array comprised of at least two switchable status identifiers associated with each of the plurality of memory blocks, and (2) which update decoder accesses and manipulates the status identifiers when applying the instruction set.

The present Office Action relies on column 5, lines 20-25 of Kulkarni in rejecting the "an update decoder resident on said client device that interprets the update package and applies the instruction set to update the plurality of blocks on a block by block basis where the flash manager (update decoder) is configured to collect data sections for the image to create the new image, and" limitation of claim 15. (Office Action, p. 10) The entire limitation is "an update decoder resident on said client device that interprets the update package and applies the instruction set to update the plurality of blocks on a block-by-block basis, and which update decoder accesses and manipulates the status identifiers when applying said instruction set". This cited column 5 passage, as discussed above with respect to claim 1, does not describe or anticipate the use of status identifiers when applying the instruction set as required by the present claim 15.

The present Office Action relies on column 5, lines 4-8 of Estakhri in rejecting the "a status array comprised of at least two switchable status identifiers associated with each of the plurality of memory blocks where an old/new flag, used free flag, and others are associated with each memory location" limitation of claim 15. (Office Action, p. 11) The cited column 5 passage of Estakhri states:

> In such circuits, a latch of volatile logic circuits is set to couple the voltage necessary to erase the FLASH cells in the block. Because of the likely large number of memory blocks in the mass

> storage 100, if the CAM 106 and mass storage 100 are on the same integrated circuit (chip).

This passage simply states that the Estakhri design includes a mechanism to erase the FLASH cells in a block by coupling the necessary voltage. No status array comprised of at least two switchable status identifiers associated with each of the plurality of memory blocks is described by Estakhri, as required by the present claim 15. Moreover, the Estakhri system and method for substituting a semiconductor mass storage device for a rotating hard disk with a focus on periodically cleaning up the mass storage device and for evenly using all blocks in the mass storage. Relevant operation of the Estakhri design is described by the passage at Col. 9 lines 16-33:

> The device includes circuitry for performing two primary algorithms and an associated hardware architecture for a semiconductor mass storage device. It will be understood that `data file` in this patent document refers to any computer file including commercial software, a user program, word processing software document, spread sheet file and the like. The first algorithm provides means for avoiding an erase-before-write cycle when writing a modified data file back onto the mass storage device. Instead, no erase is performed and the modified data file is written onto an empty portion of the mass storage. In addition, the second algorithm prevents any portion of the mass storage from being erased a substantially larger number of times than any other portion. This prevents any one block of the mass storage from failing and becoming unusable earlier than any other block thereby extending the life of the entire mass storage.

Estakhri design does not address, disclose, nor anticipates updating on a client a first image stored across a plurality of memory blocks of a non-volatile memory device to create a second image.

The present Office Action relies on column 7, lines 1-25 of Estakhri in rejecting the "which update decoder accesses and manipulates the status identifiers

when applying said instruction set" limitation of claim 15. (Office Action, p. 11). Claim 15 requires, "an update decoder resident on said client device that interprets the update package and applies the instruction set to update the plurality of blocks on a block-by-block basis, and which update decoder accesses and manipulates the status identifiers when applying said instruction set". The cited column 7 passage of Estakhri states:

> In this way, sections of the mass storage which have been erased numerous times are programmed with a reallocated data file which is rarely changed thereby allowing all sections of the mass storage to eventually approach parity of erase cycles. Like the multi-sector erase, a clean-out erase can be performed in the event that there is insufficient available storage for a data file presently being performed. For example, if all but two blocks have their respective erase inhibit flags set, and a three or more block data file is being programmed, a clean-out erase can be performed to provide sufficient storage for the data file.
>
> Once the erase inhibit flag is set for all the blocks, indicating that all the blocks have achieved parity in erase cycles, the erase inhibit and erase count registers are erased and the cycle is repeated. The selection of the maximum count depends upon the system requirements. *As the value for the maximum count increases, the disparity between erase count cycles of various blocks can also increase. However, because data is shifted as a result of achieving maximum erase count this process of smoothing cycles throughout the mass storage itself introduces additional erase cycles because a block of information is transferred from a physical block having few erases to a block having the maximum number of erases.* Accordingly, though low maximum count values reduce the disparity between erase cycles among the blocks it also increases the number of erase cycles to which the blocks are subjected.

(Emphasis added)

Thus the Estakhri design includes a mechanism determine the number of times each memory block has been erased to ensure even use of the mass storage FLASH device. No update decoder resident on a client device that interprets the update package is described by Estakhri, as required by the present claim 15.

Moreover, the Office Action asserts that Kulkarni and Estakhri are analogous arts as they both in the same field of endeavor, flash memory. (Office Action, page 11). Applicant contends that the Kulkarni art relates to updating and image in flash memory, whereas Estakhri teaches replacing a rotating hard drive mass storage memory with FLASH memory, two materially different fields of endeavor. The Office Action cites the Estakhri passage at Col.1, lines 34-43:

> Solid state memory is an ideal choice for replacing a hard disk drive for mass storage because it can resolve the problems cited above. Potential solutions have been proposed for replacing a hard disk drive with a semiconductor memory. For such a system to be truly useful, the memory must be non-volatile and alterable. The inventors have determined that FLASH memory is preferred for such a replacement. It should be noted that E.sup.2 PROM is also suitable as a replacement for a hard disk drive but it has lower performance.

Further, the Office Action cites the Estakhri passage at Col. 2, lines 60-61:

> In addition, the second algorithm prevents any portion of the mass storage from being erased a substantially larger number of times than any other portion.

In addition to these two references being in materially different fields of endeavor, neither of the above Estakhri passages includes the limitation of "an update decoder resident on said client device that interprets the update package and applies the instruction set to update the plurality of blocks on a block-by-block basis", one element is missing from the cited references. Kulkarni fails to teach a status array

comprised of at least two swtichable identifies as noted by the Examiner on page 11 of the present office action. Absence of an element from the cited references indicates claim 1 in nonobvious based on these references.

Applicants also submit that there is no motivation to combine the incremental updating of an image in FLASH memory system of Kulkarni with the even erase cycle management of FLASH mass storage memory aspects recited in Estakhri. The motivation to combine is alleged to be "to incorporate the process of avoiding erasure cycles as taught by Estakhri into the flash memory of Kulkarni". (Office Action, p. 11). This alleged motivation reads more into either reference than is present in the references themselves, and merely states a broad end result rather than a motivation to combine the teachings of the Kulkarni and Estakhri references. In reality, there is no suggestion in Kulkarni to use a status array and switchable identifiers, nor any suggestion in Estakhri to employ a difference file and a flash manager resident on a client device for creating a second image in the manner presented in Kulkarni.

In addition to the differences between the claims and the references discussed above, Applicants further dispute the combination of Kulkarni and Estakhri in the manner suggested in the Office Action. Applicants respectfully submit that no motivation to combine the references in the manner suggested is presented within the references themselves. The statement in the Office Action regarding the motivation to combine being for the purpose of "lengthen the life of the flash memory by lessening the number of erasure cycles performed during use" does not demonstrate any motivation to combine the references explicit or implied within the two references themselves, but instead takes the extending the life of the entire mass storage aspect of Estakhri and alleges that it could be applied somehow to Kulkarni. Applicants contend that simply no motivation exists to employ the memory life extending aspect of Estakhri in Kulkarni, and certainly no motivation to employ the incremental updating of an image in FLASH memory of a client device system of Kulkarni is included in Estakhri.

The Federal Circuit has held that obviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching, suggestion or incentive supporting the combination. *ACS Hospital System,*

Inc. v. Montefiore Hospital, 732 F.2d 1572 (Fed. Cir. 1984). Without some showing in the prior art that suggests in some way a combination in order to arrive at the claimed invention, it is impermissible to use the Applicants' teaching to search references for the claimed elements and combine them as claimed. *In Re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991); *In Re Laskowski*, 871 F.2d 115, 117 (Fed. Cir. 1989); *see also, Ex Parte Lange*, 72 U.S.P.Q. 90, 91 (C.C.P.A. 1947) ("It seems to us that the Examiner is using appellant's disclosure for the suggestion of the combination since there is no suggestion in any of the patents for their combination in the manner claimed by Applicant."); *In re Leonor*, 158 U.S.P.Q. 20, 21 (C.C.P.A. 1968) (the issue is "whether teachings of prior art would, of themselves, and without benefit of applicant's disclosure, suggest [a process] which would make claimed invention obvious...") (emphasis in original). As noted, the Estakhri reference does not suggest combining the mass storage FLASH memory life extending design disclosed with the incremental updating of an image on a client device aspect of Kulkarni to produce the unique method claimed in Applicants' independent claim 15.

Applicants respectfully submit that the Office Action uses hindsight in rejecting claim 15. It is only through hindsight, after seeing Applicants' disclosure, that it would be considered possible to create the reliable updating system as claimed by the Applicants. With regard to the use of hindsight, or the use of an Applicant's teaching to combine references, the courts have overwhelmingly condemned such combinations and have upheld the validity of patents or claims of patents in which such hindsight was employed to combine the references. *W.L. Gore Associates, Inc. v. Garlock, Inc.*, 220 U.S.P.Q. 303, 313 (Fed. Cir. 1983), (condemning the "insidious effect of a hindsight syndrome wherein that which only the inventor taught is used against its teacher"); *In re Fine*, 837 F.2d 1044, 1051 (Fed. Cir. 1988) ("One cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention.") Applicants respectfully submit that combination of aspects of the Estakhri reference with the Kulkarni design is merely a hindsight reconstruction of the invention using Applicants' disclosure and attempting to use Applicants' claims as a guide. Such hindsight reconstruction of the claimed system is inappropriate and thus rejection of the independent claim 15 for this reason is improper.

Based upon the totality of the foregoing, Applicants respectfully submit that claim 15 is allowable over the references of record, and that all claims dependent from claim 15 are allowable as they depend from an allowable base claim.

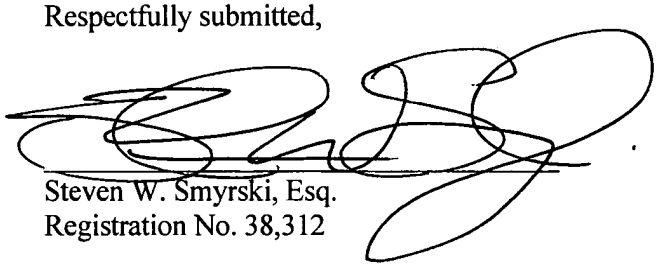
Accordingly, it is respectfully submitted that all pending claims fully comply with 35 U.S.C. § 103.

CONCLUSION

In view of the foregoing, it is respectfully submitted that all claims of the present application are in condition for allowance. Reexamination and reconsideration of all of the claims are respectfully requested, and allowance of all the claims at an early date is solicited.

Should it be determined for any reason an insufficient fee has been paid, please charge any insufficiency to ensure consideration and allowance of this application to Deposit Account 502026.

Respectfully submitted,



Date: February 23, 2006

Steven W. Smyrski, Esq.
Registration No. 38,312

SMYRSKI LAW GROUP, A PROFESSIONAL CORPORATION
3310 Airport Avenue, SW
Santa Monica, California 90405-6118
Phone: 310.397.9118
Fax: 310.397.9158